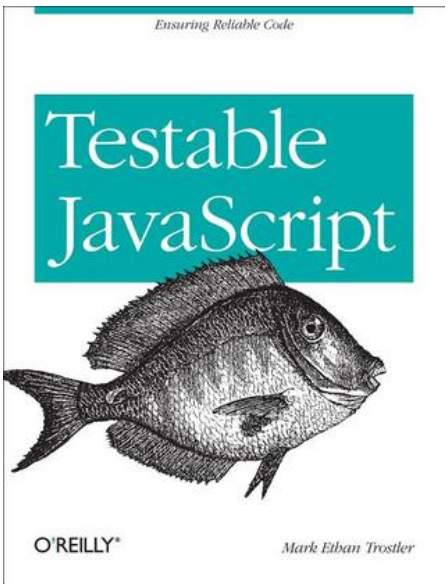

Testable JavaScript

Mark Ethan Trostler

Testable JavaScript

Mark Ethan Trostler

Testable JavaScript Mark Ethan Trostler
One skill that’s essential for any professional JavaScript developer is the ability to write testable code. This book shows you what writing and maintaining testable JavaScript for the client- or server-side actually entails, whether you’re creating a new application or rewriting legacy code.

From methods to reduce code complexity to unit testing, code coverage, debugging, and automation, you’ll learn a holistic approach for writing JavaScript code that you and your colleagues can easily fix and maintain going forward. Testing JavaScript code is complicated. This book helps you simply the process considerably.

Get an overview of Agile, test-driven development, and behavior-driven development
Use patterns from static languages and standards-based JavaScript to reduce code complexity
Learn the advantages of event-based architectures, including modularity, loose coupling, and reusability
Explore tools for writing and running unit tests at the functional and application level
Generate code coverage to measure the scope and effectiveness of your tests
Conduct integration, performance, and load testing, using Selenium or CasperJS
Use tools for in-browser, Node.js, mobile, and production debugging
Understand what, when, and how to automate your development processes

Testable JavaScript Details

Date	:	Published January 31st 2013 by O'Reilly Media (first published September 22nd 2012)
ISBN	:	9781449323394
Author	:	Mark Ethan Trostler
Format	:	Paperback 274 pages
Genre	:	Computer Science, Programming, Science, Technology, Software, Internet, Web, Technical

From Reader Review Testable JavaScript for online ebook

Naoise says

This is an interesting read.
But it's also a bit of a Frankenstein monster: for starters, the title is somehow misleading. There are bits and pieces of different, somehow-related stuff, very specific code examples, exhaustive software configuration walkthroughs, but then also philosophical dissertations and tips on good coding practices. Testing is more of a *leit-motiv*. It does have a chapter dedicated to testing.
Also, it is highly outdated. Some sections are better learnt by going to the latest up-to-date version of the software and going through the Getting Started or watching a video or reading a tutorial.
In all, if you skim through the outdated or uninteresting parts it is a fast read that stresses on good coding practices, with testing as a central point.

BCS says

JavaScript, once confined to the browser, is now finding wider application domains, particularly with the introduction of server-side frameworks such as Node.js. This book is concerned with techniques that support the development and test of reliable and maintainable software using JavaScript.

The ‘development’ theme of the book considers key strategies for writing JavaScript code that is concise, testable and maintainable.

These generally well known ideas, including the reduction of complexity and coupling within code structure, and the enhancement of source code through standardisation and documentation, are placed into a JavaScript context.

The discussion includes a description of event-based programming and event hub-based architectures as a method of promoting highly cohesive, loosely coupled software with the corresponding improvements to test and deployment activities.

The book also considers tools and techniques that help in debugging JavaScript software running in a range of client and server-side environments.

The ‘test’ theme is largely concerned with approaches to automated software testing. Here the reader is offered practical advice on the development of effective unit test cases and the use of test automation frameworks such as Selenium, Phantom.js and Jasmine.

Since the value of unit tests is correlated to the amount of code that is tested, there is a detailed discussion around the generation and validity of code coverage metrics. The book also introduces tools and techniques for integration, performance and load testing with respect to web applications. Finally the author considers tools and techniques that help to implement automated build and test processes.

Overall the book provides a useful critical survey of current methods and tools that are available to JavaScript developers. The book doesn’t really consider approaches such as Test- or Behaviour-Driven

Development, but rather presents material that could be used with these or other practices.

While not delving especially deep into any particular technique or tool, the book provides an awareness of the availability and key capabilities of salient technologies, along with practical introductions allowing the reader to get started.

The author has an easy, readable style that is at once informal and authoritative. Each chapter has useful introductions and summaries and the content is well signposted. The text is enhanced with plenty of illustrative code fragments, diagrams and links to online resources.

While the book is not really aimed at JavaScript beginners, there is a wealth of information here which should be of interest to JavaScript developers and testers.

Reviewed by Patrick Hill CEng MBCS CITP

Nephi says

This book has good information in it regarding testable code, but mostly it was about setting up environments.

Kapil Kaisare says

Mark Trostler's explanation of code metrics (like cyclomatic complexity, fan in/fan out, etc..) using Javascript makes this work worth reading. I was not impressed with his coverage of TDD and BDD methodologies, though.

Väinö Leppänen says

An excellent introduction to professional JavaScript development.

Alessandro Pellizzari says

Abbandonato circa a metà.
Non è né carne né pesce. Inizia spiegando a grandi linee i principi del codice testabile, ma nel farlo introduce concetti e codice incomprensibili per chi non sa già scrivere codice testabile.
Ci sono manuali che spiegano molto meglio i principi, e il codice non si capisce se non si conosce già il framework per il testing che usa (un componente di YUI).

Nei capitoli successivi tenta di spiegare gli Unit Test, ma non spiega mai la sintassi del framework di test. Prosegue con la code coverage con lo stesso stile. E qui mi sono fermato perché era codice incomprensibile.

Sembra un grosso catalogo pubblicitario di diversi sistemi di testing, con continui rimandi ai loro siti per

studiare la sintassi o l'installazione, e nessuna spiegazione utile.

Se non sapete niente di testing lasciatelo sullo scaffale, non imparerete quasi nulla.
Se conoscete già il framework per unit-test di YUI, Jasmine, Selenium, ecc. probabilmente qui non troverete niente di nuovo.
Onestamente non saprei a che pubblico consigliarlo.

Georgi says

The title is a little bit deceptive, as the book focuses on testing and continuous integration techniques instead of actual software design. Still, the average web developer can find some new technologies and inspiration to adopt them in his workflow.

Amp Tanawat says

Good examples, cover wide-range of test levels: unit, integration, performance, and tools.
Also have two chapters about testing practices(actually I want more :P)

Ulina says

3.7

Even though some of the tools and libraries are no longer used or outdated, the theories the author proposes are still very useful. He talks about writing testable code; writing unit tests; when to use unit tests and when to rely on integration tests; linting; and automating builds for continuous integration. He does go heavily into using YUI libraries, which are no longer used by most developers, so you can skip over those parts.

The moral of the story is, when you buy a book, read it right away, especially if it is a tech book, so the information is not so out of date when you finally read it.

Rob says

Right off the bat: I read an early access edition, and in a lot of places... it shows. It'll come through editing and be a lot tighter by the time you read it. I promise.

Putting that aside, I had some mixed feelings about this book. I'll try to explain it as follows:

Rebecca Murphey did a talk during the 2012 JavaScript Summit (put on by E4H) which was titled "Writing Testable JavaScript". Early on in that talk, she impressed upon us that she was going to talk about how to write *testable* JavaScript, and not how to write JavaScript *tests*, nor how to *test* JavaScript. These are important distinctions. Writing *tests* (in a way) is this weird hybrid of somehow both writing more code and

writing documentation. And *testing* your code is ... well, it's the act of putting your code through the accompanying tests to see if they pass muster. But writing *testable* code is about adopting a certain style wherein your code becomes easier to test. And the argument goes that if your code is easier to test, then it is also easier to reason about. (Insert functional programming rant here.)

Anyway...

Trostler talks about this, but not nearly at the length I was hoping for. Whereas Murphey kept her talk (albeit, it *was* a one hour talk) focused on adopting testable styles, Trostler's book manages to be about writing testable code, writing tests, and running tests. Given the title, I was expecting more about that functional style, more about decomposing functions into discrete units, more about how to reason about problems etc. And it wasn't even that this was *missing*, just that he was more/less finished talking about that by (say...) Chapter 3.

From there, he takes us on a tour of the rest of Test Town: writing unit tests, writing integration tests, code coverage, performance testing, load testing, debugging, linting and code quality, and how to bundle it all up and automate it. Somehow this is both too much information and not nearly enough at the same time. For example: there is a whole chapter dedicated to code coverage, and yet I walked away from it not really sure if I knew how to do that kind of instrumentation, nor if I fully understood the value. For example: integration testing, performance testing, and load testing are all jammed in to one chapter, and should probably have been split up in to separate chapters, and/but he also starts talking about things (e.g., generating HAR reports) which are... not strictly testing JavaScript.

I think maybe that's where I got my first real taste of those mixed feelings. I've been looking at a lot of front-end performance stuff lately, and so I'm very interested in it. But talking about "wire weight" is not what I think about when I see a book titled *Testable JavaScript*.

At any rate, lest this turns in to an off-color and off-topic rant: I reconcile my mixed feelings about the book in a mostly positive manner. The things that are outside the scope that's implied by the title are all things that any front-end engineer worth her semicolons should be fussing about anyway -- so that's all well and good. And I see here in my notes a snarky remark about "suggested alternative title: *Testing JavaScript with Y.Test*" -- but that's a little unfair of me. Overall, this book (and books like it) need to be written for front-end engineers. We need this badly, and I'm happy to start seeing these books crop up. This is the sort of thing that we need to see more of in our discipline.

My hat is off to Trostler for furthering the conversation; and/but: his editor has some work ahead of them both.
